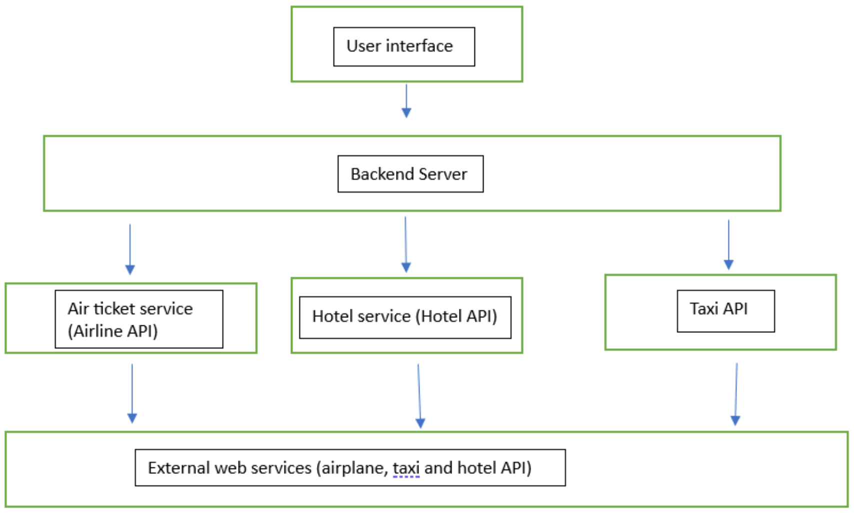
Information Technology
INTE 22212 – Software Design Patterns and Frameworks
Answers

QUESTION 1

(25 Marks)

You are asked to develop a website for travelers to book their foreign trips. Travelers (users) need to book a air ticket, hotel, and taxi. When the user requests an air ticket, a hotel, and a taxi, it should collect data from different web services and send details to the user. The Hotel sub-system will connect to various web services provided by the different hotels (third party). Similarly, the Air tickets sub-service will connect to the airline web services, and the Taxi sub-service will connect to the taxi web services. Please note that third-party organizations have developed these web services; hence web service method signatures can be different from each other.

1.1 Draw a diagram to depict the above scenarios. **[5 Marks]**


User interface
Backend Server
Air ticket service (Airline API)
Hotel service (Hotel API)
Taxi API
External web services (airplane, taxi and hotel API)

1.2 What design pattern can be employed for sub-systems that combine air tickets, hotels, andtaxis? Explain the reasons for your selection. **[4 Marks]**

The design pattern that can be employed for sub-systems that combine air tickets, hotels, and taxis is the **Facade Pattern**. The Facade Pattern provides a unified interface to a set of interfaces in a subsystem, making it easier to use and reducing the dependencies between the client code and the individual components of the subsystem. Air Ticket Service, Hotel Service, and Taxi Service can each be encapsulated by a facade, providing a unified and simplified interface for the overall travel booking process. The Facade Pattern enhances maintainability, readability, and flexibility in the design, which are crucial aspects in a system that interacts with multiple external services.

1.2.1 What challenges will you face when you design the system with the selected designpattern? **[3 Marks]**

Complexity of the Facade: Designing a simple and intuitive Facade that hides the complexities of all three sub-systems without becoming a "god object" with too much responsibility is crucial.
API changes: If the external APIs change their formats or methods, the Facade might need significant updates to adapt, requiring additional maintenance effort.
Limited flexibility: While simplifying user interaction, the Facade might offer less flexibility for customized booking workflows compared to other patterns.

1.2.2 Explain the mechanism that you will use to overcome those challenges? **[3 Marks]**

Maintain a clean interface: Keep the Facade focused on high-level functionalities and delegate complex operations to the individual sub-systems.
Utilize adapter and wrapper classes: Adapt any changes in external APIs through dedicated adapter or wrapper classes, minimizing impact on the core Facade logic.
Offer extension points: Allow for future flexibility by designing the Facade with extension points where additional sub-systems or customization options can be integrated.

1. 3 What design patterns can be employed for connecting the sub-system with different third-party web services? **[4 Marks]**
Adapter Pattern
proxy Pattern
Template method Pattern
Facade Pattern

1.3.1 What challenges will you face when you design the system with the selected designpattern? **[3 Marks]**
Selected design pattern→ Proxy design pattern
Challenges,
Introducing proxy classes may add a layer of abstraction, potentially leading to a slight increase in latency and performance overhead.
In a multi-threaded environment, managing concurrent access to shared resources can be challenging.
Introducing proxy classes may increase the overall complexity of the system.

1.3.2 Explain the mechanism that you will use to overcome those challenges? **[3 Marks]**

Increased Complexity:
Mechanism: Clear Documentation and Design Guidelines
Explanation: Thoroughly document the proxy classes, their responsibilities, and the reasons for their introduction. Establish clear design guidelines to ensure that the use of proxy classes aligns with the overall architecture and does not unnecessarily increase complexity.

Latency and Performance Overhead:
Mechanism: Optimization Techniques
Explanation: Optimize the proxy implementation to minimize unnecessary overhead. Use techniques such as lazy loading, caching, and batching to reduce the impact on performance. Profile the system to identify and address specific bottlenecks.

Synchronization Issues:
Mechanism: Thread-Safe Design and Locking Mechanisms
Explanation: Design the proxy classes to be thread-safe, especially if they involve shared resources or mutable state. Implement appropriate locking mechanisms to manage concurrent access and prevent race conditions.

QUESTION 2

(25 Marks)

A Supermarket has different types of pay calculations for different employee categories. The supermarket consists of executive employees, temporary staff, and daily workers. Following are the main salary calculations for each employee category.

Executive Employees: Basic Salary, Bonus & Executive Allowances.

Daily workers: 8 hours per day, and additional hours will be entitled to overtime. More than 200 hours will earn an attendance allowance.

Temporary staff: This category of staff is called when required, and a daily allowance will be paid for the requested dates.

All employees are contributed to Employee Provided Funds by 8%.

2.1 Draw a diagram to depict the above scenario. **[4 Marks]**

```
+-----------------------+     +-----------------------+     +-----------------------+
| Executive Employees   |     |   Daily Workers       |     |   Temporary Staff     |
|-----------------------|     |-----------------------|     |-----------------------|
| Basic Salary          |     | 8 hours per day       |     | Daily Allowance       |
| Bonus                 |     | Overtime              |     |                       |
| Executive Allowances  |     | Attendance Allowance  |     |                       |
| Employee Provided     |     | Employee Provided     |     | Employee Provided     |
| Funds (8%)            |     | Funds (8%)            |     | Funds (8%)            |
+-----------------------+     +-----------------------+     +-----------------------+
```

2.2 What design pattern(s) can be employed for the above system? Explain the reasons for your selection. **[6 Marks]**

Factory Pattern to create instances of different employee categories.

Template Method Pattern to structure the salary calculation algorithm.

Decorator Pattern to add optional features to the salary calculation process.

2.3 Draw a class diagram for the above design pattern. **[5 Marks]**

Decorator design pattern

```
+------------------------+        +------------------------+         +------------------------+
|        Component       |        |        Decorator       |         |       ConcreteComponent|
+------------------------+        +------------------------+         +------------------------+
| +calculateSalary()     |        | -component: Component  |          | +calculateSalary()     |
| +contributeToFunds()   |        |                        |         | +contributeToFunds()   |
+------------------------+        | +calculateSalary()     |          +------------------------+
        |                              +------------------------+
        |
+-------+---------+
|                 |
| Executive       |
| +calculateSalary()|
| +contributeToFunds()|
|                 |
+-----------------+
        |
+-------+---------+
|                 |
| DailyWorker     |
| +calculateSalary()|
| +contributeToFunds()|
|                 |
+-----------------+
        |
+-------+---------+
|                 |
| TemporaryStaff  |
| +calculateSalary()|
| +contributeToFunds()|
|                 |
+-----------------+
```

2.4 What challenges will you face when you design the selected design pattern? **[3 Marks]**

Maintaining Order of Decorators

Clearly defining the responsibilities of each decorator is challenging.

Testing all combinations of decorators and ensuring that they work together seamlessly is challenging.

2.5 Explain the mechanism that you will use to overcome those challenges? **[Marks]**

Clearly document the intended order of applying decorators.

Ensure that each decorator adheres to the Single Responsibility Principle. A decorator should have one responsibility and should not be responsible for multiple concerns.

Conduct integration tests to verify that the decorators work seamlessly together.

2.6 “Code duplication can be achieved from the used design pattern in the above scenario.” Do you agree with the above statement? Explain. **[4 Marks]**

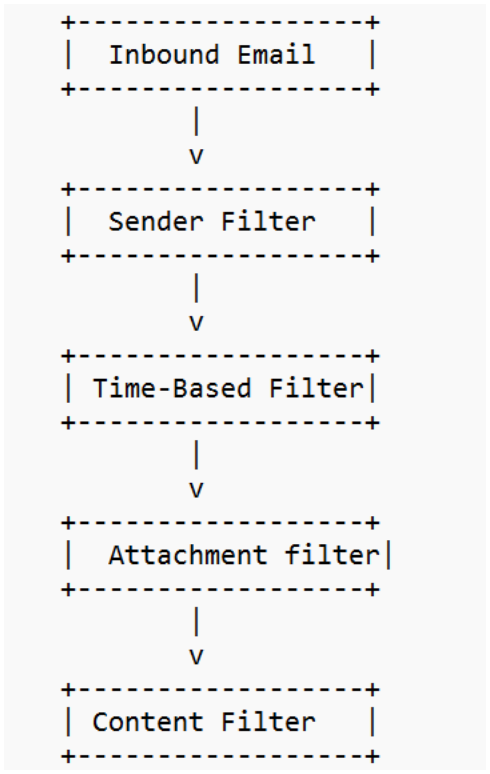
Can occur.
Code duplication in the Decorator Design Pattern can occur if similar behavior is implemented across multiple decorators. To minimize duplication, use a base decorator class, adhere to the Single Responsibility Principle, factor out common code, enforce consistent naming conventions, and conduct regular code reviews.

QUESTION 3

(25 Marks)

You are hired to design an email spam filter. In the spam filter, multiple sub-spam filters are available depending on the different components of the email, such as email sender, email sent time, email content, and email attachment.

3.1 Sketch a diagram and explain the different components of the email spam filter. **(4 Marks)**



```
+------------------+
|  Inbound Email   |
+------------------+
        |
        v
+------------------+
|  Sender Filter   |
+------------------+
        |
        v
+------------------+
| Time-Based Filter|
+------------------+
        |
        v
+------------------+
|  Attachment filter|
+------------------+
        |
        v
+------------------+
| Content Filter   |
+------------------+
```

3.2 What design pattern can you follow to design the above system? Explain the reasons. **(5 Marks)**

The design of the email spam filter involves multiple components that need to work together seamlessly to analyze and filter incoming emails. A design pattern that can be useful for structuring and organizing such a system is the **Chain of Responsibility pattern**.

3.3 How do you prioritize different components of the above email spam filter? Explain your prioritization criteria. **(6 Marks)**

User Feedback Mechanism:

Priority Reasoning: User feedback is vital for continuously improving the filter's accuracy. If users identify false positives or negatives, their feedback can be used to adapt and fine-tune the filter.

3.4 What would be your suggestion if adding more sub-components to the system is needed? **(5 Marks)**

If we need to add more sub-components to the email spam filter system, we should consider the extensibility and maintainability of the design. Before adding new sub-components, assess the current system's architecture to understand how easily it can accommodate additional modules. Chain of Responsibility pattern can smoothly integrate new components.

3.5 How you can extend this design to achieve High Availability (HA) and Disaster Recovery (DR). **(5 Marks)**

High Availability (HA):
Implement load balancing to distribute incoming email traffic evenly across multiple servers.
Design the system with a distributed architecture to allow components to run on separate servers.
Use microservices or containerization to isolate and scale individual components independently.
Employ database replication to create copies of critical data across multiple servers.
Disaster Recovery (DR):
Regularly back up critical data and configurations to an offsite location.
Maintain comprehensive documentation outlining the DR plan, including step-by-step procedures for recovery.

QUESTION 4

(5 X 5 = 25 Marks)

Explain the possible design pattern you can suggest for the following scenarios.

4.1 You are chosen to develop a website for an online cake ordering system. Users can select multiple options such as decorations of the cake and colour of the cake. Depending on the user selections, prices and delivery times can differ.

Factory Pattern: Utilize a factory to create different cake objects based on user selections. Each cake type can have its own factory, enabling easy extension for new variations. This accommodates the dynamic creation of cakes with varying decorations, colors, prices, and delivery times.

4.2 You are asked to develop a scalable and extensible product-based framework for future needs of the above online cake ordering system.

Facade Pattern: Create a facade that provides a unified interface to a set of interfaces in the system. This simplifies the complexity of the underlying system and provides a single-entry point for future expansions, making the framework scalable and extensible. The facade shields the client (ordering system) from the intricacies of the subsystems, allowing seamless integration of new features.

4.3 You are asked to develop a payroll system. The employee salary calculation should be run only by one user, and all the other connections to the salary systems should be stopped.

Proxy Pattern: Implement a proxy to control access to the payroll system. The proxy acts as a surrogate, allowing only the authorized user to execute the salary calculation while preventing other users from accessing the system. This ensures a single point of control and secures the payroll functionality.

4.4 You are developing a learning management system (LMS), and when users read a large content, it is required to load only a couple of pages of the documents rather than the entire document.

Adapter Pattern: Use an adapter to convert the interface of the document reader component into one that the LMS expects. The adapter enables the LMS to load only a few pages of the document at a time, seamlessly integrating the partial loading functionality without modifying the existing document reader.

4.5 You are developing an LMS that needs different modules to be created depending on the user's needs.
Factory Method Pattern: Implement a factory method for module creation, allowing each user type to have its own factory. Users can request the creation of modules, and the appropriate factory will generate the specific module based on their needs, ensuring flexibility and easy extension for future user requirements.

Note.
Answers are taken from ChatGPT and bard.